Hierarchical spatial-temporal state machine for vehicle instrument cluster manufacturing

Tomasz Maniak, Rahat Iqbal, and Faiyaz Doctor

Abstract— **The vehicle instrument cluster is one of the most advanced and complicated electronic embedded control systems used in modern vehicles providing a driver with an interface to control and determine the status of the vehicle. In this paper, we develop a novel hybrid approach called Hierarchical Spatial-Temporal State Machine (HSTSM). The approach addresses a problem of spatial-temporal inference in complex dynamic systems. It is based on a memory-prediction framework and Deep Neural Networks (DNN) which is used for fault detection and isolation in automatic inspection and manufacturing of vehicle instrument cluster. The technique has been compared with existing methods namely rule-based, template-based, Bayesian, restricted Boltzmann machine and hierarchical temporal memory methods. Results show that the proposed approach can successfully diagnose and locate multiple classes of faults under real-time working conditions.**

Index Terms—**Spatial-temporal inference, fault detection, fault isolation, neural networks, Restricted Boltzmann Machine, hierarchical spatial-temporal state machine, deep belief network, data analysis.**

I. Introduction

Instrument Clusters (ICs) serves as the main interface to display a multitude of driver information and provide assistance systems within a vehicle. ICs like many other vehicle components are manufactured with the use of computer-based manufacturing systems. Such systems are equipped with a variety of different devices and control systems necessary to manufacture products capable of satisfying customer requirements. A modern computer based manufacturing system consists of a number of manufacturing cells performing a range of assembly operations and functional tests. The cells are controlled by custom built software for supervising a given production process.

One of the most important tasks assigned to computers supervising manufacturing plants is to detect and diagnose product faults. The most common forms of fault detection used in many manufacturing plants include limit checking which is done using Statistical Process Control (SPC) [1]. SPC is simple, robust and reliable but slow to react to changes in a process characteristic. SPC fails to identify complex faults only found by observing patterns in data and how these patterns change over time. This approach disregards complex data patterns resulting from the spatial-temporal nature of manufacturing data limiting their usefulness and fault detection capabilities.

Artificial Neural Networks (ANNs) have proven to be successful in achieving good performance on tasks such as image and speech recognition [2], [3]. ANN is a network of neurons, which learns very complex functions through a series of nonlinear transformation. Recent discoveries of deep learning techniques [4] can more efficiently learn and generalize to new data points. ANNs have been adopted to address fault diagnosis problem [5]. However, most previous work utilizes shallow neural networks concentrating mainly on supervised learning to achieve the Fault Detection and Isolation (FDI) functionality. Thus, existing methods cannot detect novel-unseen fault types.

Hierarchical Temporal Memory (HTM) is a successful algorithm based on recent discoveries in neuroscience. HTM is a machine learning model inspired by the mammalian neo-cortex consisting of hierarchically connected nodes where each node represents a simplified model of a single group of cortical columns [6]. It contrasts with more simplistic and mathematically understood implementations widely used in most ANN models. HTM has been successfully applied to several complex problems such as license plate recognition [7].

This paper presents a novel approach to the problem of FDI in automatic computer-based inspection systems. The approach can identify correlations between spatial-temporal sequences of input patterns based on prediction based learning and optimisation aimed to minimise the distance between predicted and actual values of an input vector. The proposed approach improves the detection, isolation and prediction capabilities of existing FDI systems, reducing costs related to performance degradation and machine downtime, improving First Time Yield (FTY), reliability and quality. The conclusions are supported by experiments performed on a real production line in order to measure the system's performance under different fault conditions including novel faults introduced to the system for the first time. The performance of the system is also compared with other widely used FDI techniques.

The proposed approach is presented in the context of manufacturing systems; however, can be potentially applied in sensor fusion [8], intelligent vehicular communications [9], [10] localization [11], mobile data communication [12], driver support systems [13], data retrieval [14] among others.

The rest of this paper is organised as follows. The literature review in Section II presents an overview of FDI methods currently used within automotive manufacturing systems. Section III describes the proposed automated inspection FDI systems and outlines the methods, system architecture and stages included in the process. Section IV presents the experiments and the results of the performance of the proposed system observed on a real production line.

Concluding remarks and future work considerations can be found in section V.

II. Literature Review

A conventional method for ensuring the fault free operation of manufacturing production lines is to periodically check the process variables. They include software configuration validation, sensor validation, measurement device calibration and preventive maintenance. This is performed in accordance with a periodic schedule and well-defined predetermined procedures. This method is not able to detect other type of faults which can only be detected by continuous assessment of variables such as incipient process faults. Owing to an increase in process complexity and sophistication of production equipment this method is no longer cost effective and in many cases impractical or impossible to implement on large scale computer based production lines [15]. That is why a significant investment has been made to develop new methods for a more systematic approach to this problem.

FDI methods can be mostly categorised into two main groups: hardware redundancy and analytical redundancy [16]. The main idea behind redundancy based methods is to generate a signal which represents a difference between the normal and actual measured behaviour of a system under inspection. By considering this signal and how it changes during the normal operation of a machine, a fault occurrence can be detected. Based on this description it can be easily inferred that hardware or parallel redundancy is based on creating a residual signal through the use of hardware implementations [17].

The general idea behind hardware redundancy method is to measure a given process variable with more than one sensor and detect a fault by performing consistency checks. This FDI technique relies on a voting scheme performing consistency checks of the redundant component in order to determine a fault occurrence and location [18].

An alternative approach to hardware redundancy is analytical redundancy. Instead of using an additional hardware a mathematical model of the monitored process is created and compared with mathematically derived relationships between the different variables of the model which are used as a reference with actual system outputs [19], [20]. Any variation between measurements of these relationships and the outputs of the system is called a residual and can constitute a failure in a system. In a fault free system, the residual signal should ideally be equal to zero. This is however seldom the case, and often even when there is no observed fault, residuals are different to zero. That situation is caused by measurement uncertainties and noise [21]. To deal with this issue a threshold on the residual signal is often set to avoid false alarms. The analytical redundancy based FDI systems consists of residual generation (necessary to create the residual signal) and residual evaluation (performed to infer the fault status of the system). Fault detection systems based on analytical redundancy are often called model based systems and do not require costly installation and maintenance of additional hardware, a clear advantage over hardware redundancy based FDI [22].

The three most common frameworks to realise these models are mathematically-based, expert-system based, and Computational Intelligence (CI) based frameworks as shown in Figure 1. In mathematically-based frameworks, residual generation is achieved by a physics-based mathematical model of the process. The framework aims to find the discrepancies between the expected behaviour of the system produced by the model and the actual system output [23].

Expert-based frameworks on the other hand use an expert knowledge of the system to recognise the symptoms and infer the state of the system [24]. Finally CI-based frameworks use historical data about the process measured by the sensors to generate the residual signal and check for process faults [25], [26].

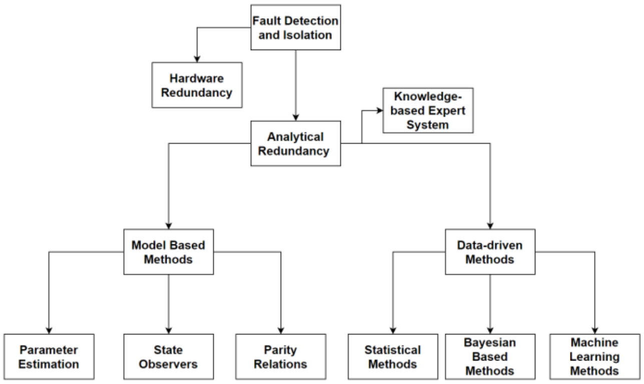

Fig. 1. Categorization of fault detection and isolation methods based on a priori knowledge.

III. Hierarchical spatial-temporal state machine (HSTSM) model for fault detection and isolation

Here we describe the novel spatial-temporal generative model for FDI called the hierarchical spatial-temporal state machine. The proposed approach integrates four soft computing techniques: Deep Belief Networks (DBN), auto-encoders, agglomerative hierarchical clustering and the n-order Markov chain. The functional steps of the HSTSM can be viewed using an information hierarchy model (see Table 1) such as DIKW (Data, Information, Knowledge, Wisdom) pyramid [27] representing the structural and functional relationship between data, information, knowledge and understanding.

TABLE I
HIERARCHICAL SPATIAL-TEMPORAL STATE MACHINE - STEPS INVOLVED

No	Method	Purpose	DIKW hierarchy component
1	Data Acquisition	Acquiring raw data from manufacturing machines.	Data
2	SPC Data Encoder	Encoding the raw data into a vector of sparsely distributed representations	Information
3	Deep belief network with Auto-encoder	Feature extraction and data compression , dimensionality reduction	Knowledge
4	Hierarchical clustering into n-classes	Spatial pooling of data	Knowledge
5	State encoding with the use of n-order Markov model	Temporal inference	Knowledge
6	Residual vector acquisition	Fault detection with the use of distance function applied to input vector and predicted vector	Understanding
7	Fault inference and isolation	Fault isolation with the use of MLP	Understanding

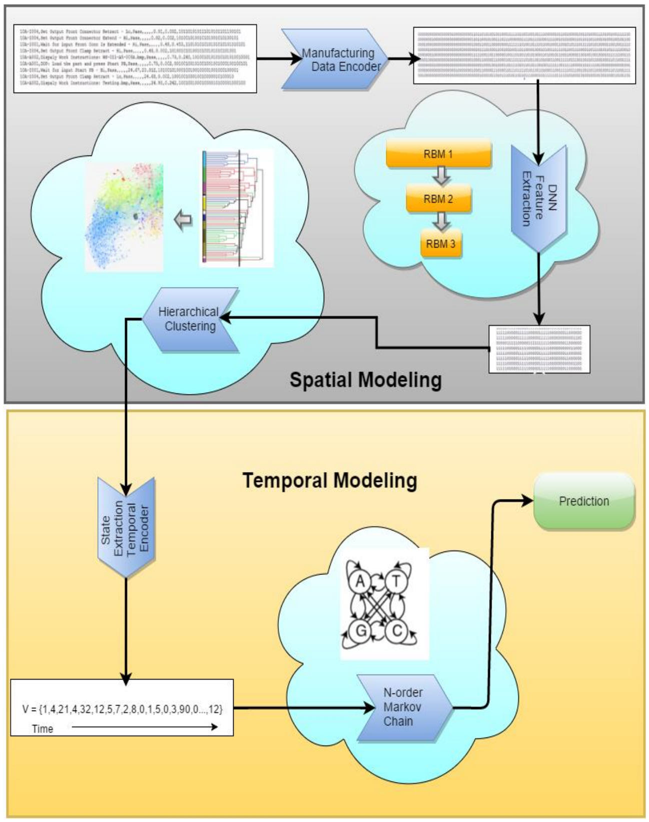

Fig. 2. Overview of proposed method.

A graphical illustration of HSTSM's steps is presented in Figure. 2, where each step is described below.9

A. Data acquisition

With respect to the real world system presented in this article the following data has been identified as the source of information about the process:

- Statistical process data generated by individual parts of the IC production process.
- Raw input data generated by various sensors monitoring the different characteristics of the IC process.
- Discrete I/O signals from digital sensors monitoring the process.
- Sequential and timing behaviour of a device (individual task time, relative performance time, test execution time, etc.)

The combination of data can be considered to represent a unique point in a multidimensional input space. The raw data is transformed into a binary vector using the SPC data encoder to build an understanding of complex relations between the inputs and outputs of the underlying process.

B. SPC Data encoder

To make meaningful use of the data acquisition process described previously, the data output needs to be specifically encoded. This encoding is used to uniquely represent symbols from one or a more source alphabets to a target alphabet. In this case the target alphabet has the following imposed constraints: the encoded strings have to be binary, sparsely distributed and have to represent different physical characteristics in a meaningful and easily interpretable way.

During manufacturing the product undergoes a number of assembly and test operations executed by a combination of manual labour and automated processes. The operations can be thought of as a set of discrete events occurring over time. Each of the events generates some associated process data. This data can be measured, monitored and logged by a number of dedicated sensors. Although this sequence of events is continuous and infinite, a unit of time t_i can be identified in which the set of events β is finite and can be discretised. By unit of time in this context we mean a measurable value which is a function of time $f(t)$. For most manufacturing processes one noticeable unit of time can be a sequence of test and assembly events which can be further divided into individual tests. The most natural way of discretising a manufacturing process would therefore be to divide it into a repeatable sequence of individual tests or operations. The great advantage of defining a unit of time in this way is that the set of tests for a particular process E is finite, and easily definable. At each $\mathrm{t_i}$ step a fixed number of parameters (signals) can be identified for this model. Let $\sum \subseteq \boldsymbol{N}$ be a set of all possible tests, $\varphi \subseteq \boldsymbol{R}$ be a set of all possible values for a corresponding test at time t_i , $T \subseteq \boldsymbol{R}$ a test time expressed in seconds as a difference between t_{i+1}- t_i and $Z \subseteq \{0,1\}^n$ be a set of all possible discrete I/O signals for a particular device before and after the test execution. An input to the encoder at time t_i is of the form (s,v,c,z) where $s \epsilon \sum$ is a test , $v \epsilon \varphi$ is a value for a particular test at time t_i, $c \epsilon T$ is a cycle time for a test, and $z_i \epsilon[z_1,z_2 ,... ,z_n]^T$ a vector of all discrete I/O signals generated by a device before and after the test execution. The main function of the encoder is to map the inputs defined above to discrete signals which can be understood by the HSTSM. Some of the parameters of the input are discrete by their nature, the remaining parameters (*s*,*v*,*c*) must be mapped to a space of discrete translations. First to be considered is test $s \epsilon \sum$.

In order to discretise value *t* into a binary vector containing $n \epsilon \boldsymbol{N}$ elements where *n* is equal to $|\sum|$, the value of i-th element representing the corresponding test is set to 1 and all other elements of that vector to 0. For $v \epsilon \varphi$ representing an output value for a corresponding test this operation is more complicated. It must be remembered that the process of discretisation always introduces an error to the data. In this method of discretisation for real valued numbers the scale of the error is a function of the resolution used. The bigger resolution used the less the error in the data and vice versa. By resolution we mean the number of intervals into which the range of a given attribute is divided. In this research 40 intervals for each value of a test were used. Each interval represents an element in a binary vector $X = (x_{1\,,} x_{2\,,} , \ldots , x_{40}) \epsilon \{0,1\}^{40}$.

The intervals for each test are calculated in the following way. A random number of samples proportional to *n* number of measurements are taken and both mean value μ and standard deviation σ are calculated. The width of the interval W is calculated as follows: $W = 1/4\ \sigma$. For each element x_i in vector *X* the binary value is set to 1 if input value $v \epsilon [\ \mu\text{-}(20\text{-}i)W, \mu\text{-}(21\text{-}i)W\ ]$. For all other elements the value is set to 0. If the value $v < \mu\text{-}20W$ then $x_1 = 1$ and $x_2 = 0,\ x_3 = 0 , \ldots , x_{40} = 0$, if $v > \mu\text{-}40W$ then $x_{40} = 1$ and $x_1 = 0,\ x_2 = 0 , \ldots , x_{39} = 0$. For all values of *c*- cycle time an encoder defined above is used to encode *c* into $C = (c_1, c_2 , \ldots , c_{40}) \epsilon \{0,1\}^{40}$. An example of an input encoded in this way can be found in Figure. 3.

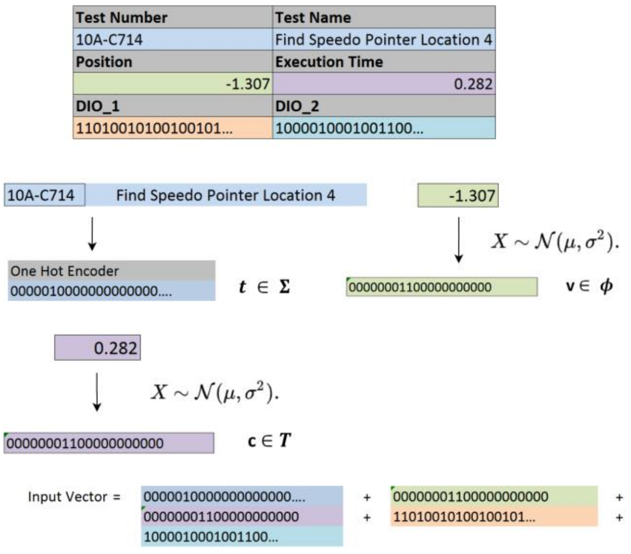

Fig.3. Data encoder.

The example shows a decomposition of a manufacturing test into individual components and their corresponding output values after encoding. The output values are concatenated to create an input vector. The example presented is simplified for easier visualization. A typical test contains greater number of digital and analogue signals.

All the individual vectors generated by the encoder are concatenated to create a binary input vector $v = (s_1, s_2 , \ldots , s_k , v_1 , v_2 , \ldots , v_n , c_1, c_2 , \ldots , c_m , z_1 , z_2 , \ldots , z_l)$ which is used as an input to the HSTSM. To handle any missing information in the dataset the encoded mean values of a given property were used, which ensures a minimal bias for the system.

C. Deep belief network with auto encoder for spatial dimensionality reduction

Consider a set of tests each of them potentially correlated with each other in a specific way. One solution to the problem of incorporating this knowledge and embedding it into the model might be to devise a number of basic rules describing those correlations. This approach of using expert knowledge to model complex technical processes observed in most manufacturing plants is often inefficient and impractical. Rules have to be manually discovered through the process of data analysis which for most complex manufacturing processes requires extensive human effort. This effort has to be repeated every time a new process is introduced, while for existing processes, every time a change is introduced it is necessary to add or amend some of the rules resulting in a constant process of supervision. In this sense it might be necessary to consider a more automated approach. Over the last decade, a number of unsupervised machine learning techniques have been discovered which successfully contributed to achieving this goal in a number of applications. If the hidden causes of correlations between different tests are considered to be represented by latent features which need to be discovered, the task can be stated in terms of a feature extraction problem. A very efficient feature extraction method for multiple levels of representation is called a DBN which is a type of a generative graphical model composed of multiple levels of hidden units. This architecture of connected hierarchical layers creates a DBN. A DBN is a composition of simpler unsupervised models where each consecutive hidden layer serves as a visible layer for the next level. For the purpose of this research an Restricted Boltzmann Machine (RBM) model is used to form the DBN. The RBM enables a binary version of factor analysis to be performed to discover latent features that explain the underlying data and find complex regularities in the training data. The use of these latent features can help to find specific faults in the system. The RBM is a stochastic neural network consisting of one layer of hidden and one layer of visible units. It is an undirected graphical generative model where each visible unit is connected to all hidden units. It models the distribution of visible variables with the use of hidden variables. RBM is based on the energy term as described in [28]. Geoffrey Hinton proposed a training algorithms called constructive divergence (CD), which is an approximation to the gradient of the log-likelihood that uses Gibbs sampler initialised at the inputs. More infromation about RBM and its training methods can be found in [29]. Although an RBM is a powerful tool, a single layer of binary features is not capable of representing the regularities in data in the most optimal way. By stacking more than one RBM in a layer wise fashion a better model can be created capable of representing features in a hierarchical manner. It was proven that by adding additional layers of RBM a lower bound can be achieved on the log probabilities that is assigned by the model to the training data, provided that all the requirements stated in [28] are fulfilled. A structure created in this way is called a DBN and its graphical representation can be found depicted in Figure 4. The structure created by stacking multiple layers of RBM is trained in the following way. First the RBM is trained on the data as an input with one hidden layer. Then the signal is processed to another layer using the hidden activations of the first RBM as the input to the second RBM, the process is repeated until the final layer is reached. In this way multiple layers of feature representations are created where each consecutive layer is a higher-level representation of the input.

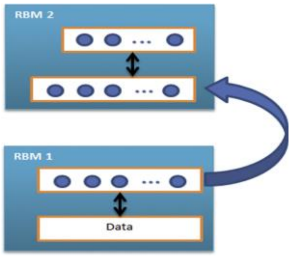

Fig. 4. Bipartite graph of RBM.

The DBN formed by the stack of RBMs creating a hybrid generative model can be used in conjunction with deep auto encoders for dimensionality reduction, to represent the data in a more compact way [30]. In this case a bottleneck has to be introduced in one of the layers. RBMs are used to pre-train the individual layers of deep auto-encoder (DAE). This approach ensures good approximations of the solution. The generated weight matrix is then used to initialize the DAE [31]. Pre-training is followed by one of the many back-propagation techniques to fine-tune the auto-encoder. This procedure allows better, more compact representation of the data to be found. This step reduces the data dimensionality and ensures that the rest of the model works with better and more compact representations of the input data.

D. *Hierarchical clustering into n-classes*

The understanding of how the different input vectors correlate to each other is critical to the problem of novelty detection and more generally unsupervised fault detection. Cluster Analysis, also called data segmentation can help to build hierarchical representations of the underlying data. This is necessary to learn how the position of the individual data points varies and using a distance function some form of correlation can be assumed between them. By gaining this understanding the difference between a new input data point and ones used during training can be assessed.

One of the main tasks in fault detection is to classify novel input data points seen for the first time by a model, considering at the same time the extent to which they differ in some respect from the points that were available during training. This task can only be successfully achieved when a good model of "normal" data can be constructed. Cluster analysis techniques can be used for constructing this kind of model and can also compensate for the insufficient amount of "abnormal" data necessary to construct explicit models for non-normal classes. In the proposed approach agglomerative hierarchical clustering [32] is used where the algorithm generates a dendrogram representing the hierarchical structure of the underlying data illustrating graphically how data points are organised in the high dimensional latent features space, generated by the DBN.

In order to reduce the flat clustering of the dendrogram (i.e. the set of clusters that without any explicit structure would relate the clusters to each other), very often a threshold is selected on the distance which "cuts" the tree on that distance. This is necessary to reduce the number of clusters and at the same time add the structure to the set of clusters. Hierarchical clustering in the context of HSTSM is performed to encode the compressed representations (expressed as hidden features of the bottleneck layer of an auto-encoder) into one unique state (which is an element of a set of all the available states representing the underlying process). The number of elements in this set is equal to the number of discovered clusters and depends on the threshold of the distance used with the acquired dendrogram. Hierarchical clustering used with HSTSM is an element that pools the spatial correlations in the input vector, representing them in one unique state from the set of possible states that the underlying process can generate.

E. *State encoding with the use of n-order Markov model*

So far, the spatial aspects of the input data (which can also be referred to using HTM terminology as spatial pooling) have been described. The use of the N-order Markov model is proposed to tackle the problem of incorporating temporal knowledge of the causal relations of manufacturing data into the proposed model. By extending the method to incorporate the N-order Markov model both spatial and temporal regularities in the underlying data can be inferred. Patterns of activities occurring both in time and space can thus be learned. In this way regularities in the sequence of events recorded from the data can be detected leading to an understanding of how the potential faults occur and help learn the temporal patterns which are generated by the underlying process. To achieve this, the multidimensional data needs to be encoded into a sequence of individual states. The clusters generated in the previous step are used explicitly for this task allowing individual data points to be encoded into one unique system state. These individual states follow one another in a sequential manner forming a Markov process. It is assumed that the underlying manufacturing process changes states according to some transition rules. The purpose of this step of HSTSM is to discover those transition rules with their probabilities and store those temporal patterns for a future inference and prediction process. The canonical probabilistic model for temporal data used within this work is called the N-order discrete Markov chain. It assumes that the future state is independent of the past state when the present state is given. The N-order Markov model is very often used when the first order Markov chain is not sufficient to comprehensively describe the transitions between states.

F. Residual vector acquisition

Let $S_{in(t-1)}$ be a previous state of the monitored process acquired by encoding an input vector $V_{in(t-1)}$ at time t - 1. The encoding is performed in two stages. First a vector $V_{in(t-1)}$ is processed through the trained DBN model to produce a vector of features $V_{dbn(t-1)}$. Next a cluster which is closest in distance to the vector $V_{dbn(t-1)}$ is selected as a previous state $S_{in(t-1)}$. By using the transition matrix of the N-order Markov model the current most likely state of the monitored process $S_{in(t)}$ can be predicted. The state $S_{in(t)}$ can be decoded in the form of a prediction vector $V_{pred(t)}$ representing a point in multidimensional feature space learned by the DBN. Consequently, if $V_{in(t)}$ is a current input vector of the monitored process, $V_{in(t)}$ can be mapped into the feature space using the DBN generating $V_{dbn(t)}$. A new vector $V_{residual}$ can be created such that $V_{residual} = V_{dbn(t)} - V_{pred(t)}$. A sum of all elements in $V_{residual}$ would be a measure of the level of discrepancy between what the model believes the status of the input to be and the actual input. This vector can be used to do a basic inference of the fault occurrence. The results presented later show that by introducing a threshold on this measure a basic classifier can be created to successfully detect faults in the system. Better results however are achieved by using the whole $V_{residual}$ where all the individual elements of the vector are used as an input to a classifier. The use of $V_{residual}$ is also necessary for isolating a failure in the system. The approach of subtracting $V_{dbn(t)}$ from $V_{pred(t)}$ helps the regularities in the data to be disregarded, in other words, ignore what the model predicted correctly and concentrate only on the discrepancies thus making the classification of a fault easier. The difference between the two vectors expresses the irregularities between the normal behaviour of the system and the actual behaviour of the system. $V_{residual}$ can be used with any supervised machine learning classification algorithms to identify the type and occurrence of a fault. In this work an ANN algorithm called multi-layer perceptron (MLP) has been used for this purpose. The MLP consists of a single hidden layer and takes as an input $V_{residual}$. For the activation function *tanh* has been used, and the network is trained with stochastic gradient descent. The objective function used isis negative log likelihood. For the multi-class classification problem, the *softmax* activation function isis used in the output layer of the neural network. This inference process is depicted in Figure. 5

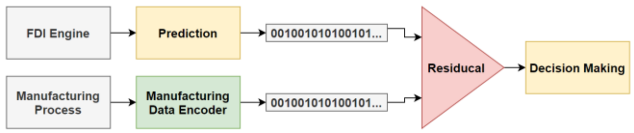

Fig. 5. Fault detection inference process.

IV. Experimental Results

The proposed approach was tested on real life production equipment performing the automated functional inspection of ICs over a period of one week. The production system used for this study was a BMW MINI F56 IC automated inspection system equipped with 57 digital input signals (connected to corresponding digital sensors), 34 digital output control signals, an automated vision system and a number of continuous signals (inspected by designated measurement equipment). All the samples had been automatically generated by the machine through a process of part inspection, calibration and machine operation. The overall process of inspection can be summarised as follows: Initially an operator loads a part into the automated inspection machine; the device recognises that the part is present and executes an appropriate test sequence which depends on the variant of the inspected part. The machine first connects to the IC using the Car Interface Network (CAN) protocol and initialises the connection between the IC and the tester. During the automated check a tester using the CAN interface sends a series of CAN messages to the IC. The messages are interpreted by special firmware in the IC's microcontroller and appropriate actions are performed by the Instrument Cluster Unit (ICU). The inspection machine using several sensors and measurement devices checks the responses to the CAN messages and individual digital signals generated by the tester. Some of the tests performed by the inspection machine include: turning indication lamps on or off, checking their correct function with the use of a camera system, checking the shape and colour of the indicator lamps, moving the indicator gauge to a number of different positions, checking the correctness of the gauge position and checking a number of different electrical characteristics of the product (e.g. the outputs of stepper motor drivers and other audio, visual, mechanical and electrical characteristics).

The data used to train the learning module was composed of 15,000 samples, divided between train (70%), validation (15%) and test (15%) datasets. The training dataset was used to determine the weights and biases of the proposed generative model. To evaluate the classification and novelty detection capability of HSTSM, the generative capabilities of the model was evaluated in a static spatial domain to ensure good reconstruction of inputs. Hence a study was conducted to check the static reconstruction of the signals for each system state, as encoded by automatically learned features, derived from the deep auto-encoder, pre-trained with RBMs. The DNN used for this study was composed of two RBMs with the following configurations: RBM 1 – 120 hidden units, learning rate 0.004; together with RBM 2 – 80 hidden units, learning rate 0.01. The training algorithm used with the RBMs was Persistent Contrastive Divergence (PCD). The hyper-parameters used for the model had been chosen using grid search optimization technique. The goal was to adjust the hyper-parameters based on a defined subset of the hyper-parameter space to find the best set of hyper-parameters that minimized the test error [33]. The error function used to evaluate the fitness of reconstructions was the logarithm of the likelihood function for a Bernoulli random distribution. Figure 6 shows how the error on reconstructions change as a function of the number of epochs used to train the deep auto-encoder with RBM pre-training.

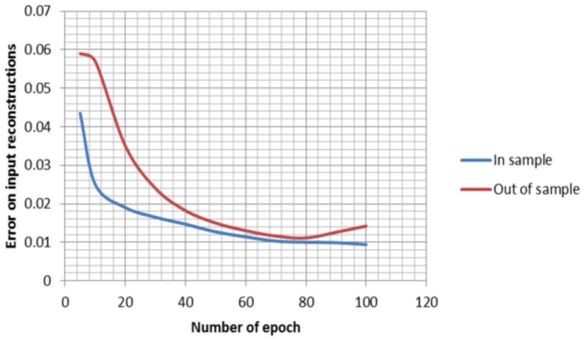

Fig. 6. Error on input reconstructions in sample vs out of sample as a function of epochs trained

This study was performed to investigate both the in-sample and out-of-sample error of the trained deep auto-encoder to ensure the model could generalise well on unseen data, meaning that the model can encode the information for novel inputs with similar accuracy. The two errors (in-sample and out-of-sample) were represented as a function of the number of epochs in a form of a chart as shown in Figure 6. The figure clearly shows that there exists a point based on the number of epochs for which the model is trained, after which the reconstruction error for training samples goes down, but the reconstruction error on the validation set increases. This is due to the problem of model overfitting [34] where a model fits the input data too closely and does not generalise well on the unseen sample data.

There are several solutions to this problem, one of them being the use of regularization techniques. Such techniques introduce additional information to the model, which prevents the selection of unreasonable parameters for the model given the context of a problem. A different approach recently suggested by [35] was the use of a technique called Dropout where the method of setting a random output of a given layer to 0, based on a given probability was implemented. Extensive experiments prove the usefulness of such a technique [36]. This technique is applied in this work and shown to improve the input reconstruction results as presented in Table II. Consequently, an application of a different technique based on learning rate adaptation called momentum was considered. The momentum value has been set to 0.9. This hyper-parameter has been chosen using grid search optimization technique [34] with the objective to minimize the test error. The influence of momentum on the error of input reconstruction is measured and presented in Table II.

TABLE II
PARAMETERS OF THE MODEL AND THEIR INFLUENCE ON INPUT RECONSTRUCTIONS

Network type		Error rate on input reconstructions
With dropout	Without momentum	0.0069
	With momentum	0.0047
Without dropout	Without momentum	0.0074
	With momentum	0.0071

This study shows a positive influence of the momentum technique on the reduction of error in input reconstruction. The next study was performed to analyse the influence of the different optimization methods on the input reconstruction and is presented in Table III.

TABLE III
SELECTED OPTIMISATION METHODS AND THEIR PERFORMANCE

Optimisation algorithm used	Error rate on input reconstructions
SGD	0.0047
Adam	0.0042
RMSprop	0.0039
Adadelta	0.0121

The model was trained using CUDA and a GeForce GTX 760 GPU device which was proven to speed up the overall computation time by a factor of 3. Training the model with the support of GPU for 100 epochs and 10,500 samples took in total 1 hour and 26 minutes. The same model configuration executed with the use of CPU took 4 hours 14 minutes to complete. The maximum execution time of the method with the above configuration on a real-time system was equal to 274 ms. The computational unit used for the study was ADLINK PXI-3980 Controller.

Table IV presents the different error rates for input reconstructions for different number of layers used both with pre-training and random weight initialised RBMs. This study was performed with the use of the validation dataset.

TABLE IV
ERROR RATE ON INPUT RECONSTRUCTION FOR DIFFERENT NUMBER OF HIDDEN LAYERS

Network		Error rate on input reconstructions
Type	Depth (Architecture)	
Deep Auto encoder + RBM pre-training	1	0.1436
	3	0.0081
	5	0.0523
Deep Auto encoder + random weight initialisation	1	0.2877
	3	0.1157
	5	0.0691

Considering the above the influence of the number of hidden units used when training the two models with three hidden layers was evaluated. Here the maximum execution time for real-time operation of the methods was measured.

The results from Table V suggest that networks consisting of 350 hidden units (across the two layers of DNN 275-75) produced the smallest error on reconstructions. Worth noticing is the fact that like the previous study the same number of hidden units in the DNN works best for both types of network (with pre-training and random weight initialization). In both cases the best results are achieved with model pre-training.

TABLE V
ERROR RATE ON INPUT RECONSTRUCTION FOR DIFFERENT NUMBER OF HIDDEN UNITS

Network			Error rate on input reconstructions
Type	Total number of hidden units in the DNN	Maximum execution time (ms)	
Deep Auto encoder + RBM pre-training	250	164	0.0126
	350	271	0.0074
	450	343	0.0089
	550	527	0.0096
Deep Auto encoder + random weight initialisation	250	164	0.2658
	350	271	0.1043
	450	343	0.3674
	550	527	0.4673

The best overall results on input reconstructions have been achieved with the following network architecture: 664 – 275 – 75 – 275 – 664. The model was pre-trained in a greedy layer wise fashion using RBMs. Dropout had been used as a regularisation technique. The learning rates used were respectively: RBM 1 = 0.006, RBM 2 = 0.02 and deep auto encoder = 0.07. Finally, the RMSProp optimisation algorithm was used to adjust network weights. The network was trained with the use of GPU for 1000 epochs and achieved the input reconstruction error of 0.0039. The reconstruction error of 0.0039 as shown later in experiments is sufficient to encode the input data and represent it as a state for the N-order Markov chain to achieve good fault detection results of up to 84% accuracy for faults in product.

In the next study an analysis of the measure of the discrepancy between $V_{dbn(t)}$ and $V_{prediction(t)}$ generated by the proposed HSTSM model was performed to assess if simple linear classification on the distance measure could be used to detect a failure in a system. The following histogram shows the sum of all the elements of $V_{residual}$ expressed as an absolute value of normalised scalar value ¥ ϵ R for selected data samples. It can be concluded with high probability from Figure 7 that there exists a distinct point around ¥ = 0.3 where for all values ¥ are bigger than 0.3 and the existence of a fault in the system can be suspected. The diagram also indicates that a simple fault detection system can be achieved with the use of HSTSM and a linear classifier. This information gives some grounds to claim that $V_{residual}$ contains useful information about a potential failure and can be used, not only to detect an occurrence of a fault, but also allows the classification of the type of fault. Considering the above an investigation into the performance of a fault detection system based on HSTSM and MLP classifier was conducted. Firstly an HSTSM with the following architecture was trained: RBM(1) 664 – 275 ; RBM(2) 275-75; AE 664-275-75-275-664.

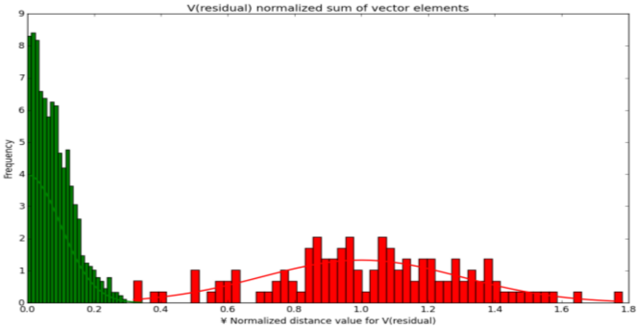

Fig. 7. Histogram of normalised scalar value ¥ acquired from vector $V_{residual.}$

The HSTSM algorithm was trained on 15000 unlabeled samples. Consequently 1200 manually labelled samples were each assigned to one of the following classes: "OK" and "NG", distributed in the ration of 60:40. The data was further divided into training data - 800 samples and validation data - 400 samples equally distributed between the two classes. Next an MLP classifier was trained with a one hidden layer 75-120-1 feed forward architecture using stochastic gradient descent. The selected cost function used for the purpose of optimisation was negative log likelihood and the transfer function - *tanh*. The inputs to the MLP classifier were all the elements from the $V_{residual}$ generated by subtracting the vector of HSTSM predictions from the actual vector of encoded input activations. To assess the performance of the classifier on a validation data a system was trained, results logged and presented in Figure 8 as a confusion matrix. From this performance matrix it can be identified that the overall accuracy of the proposed classifier is equal to 98%. Considering the calculated accuracy of 98% the 2% misclassification rate can be inferred. Other indicators derived from the confusion matrix are: the sensitivity of 97.9%, false positive rate - 1.8%, precision - 98.7% and Kappa indicator 0.958.

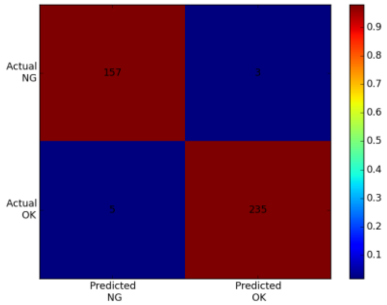

Fig. 8. Confusion matrix for fault detection and isolation with HSTSM + MPL

In order to perform more sophisticated fault isolation and identification the MLP classifier was trained with four unit *SoftMax* output layers each of them corresponding to one of the following classes: no fault, faults in a product, faults in inspection equipment, configuration faults.

TABLE VI

PERCENTAGE OF CORRECTLY CLASSIFIED FAULTS FOR DIFFERENT FAULT CLASSES WITH DIFFERENT FAULT DETECTION METHODS

Method	DSTM (proposed method)	HTM+RBM	Template based method	Rule-based method	Rule-based method	Bayesian based method
Faults in product	84%	81%	61%	48%	48%	72%
Faults in equipment	72%	68%	59%	51%	51%	57%
Configurat ion fault	59%	46%	N/A	36%	36%	N/A

The product quality defects defined here as faults in a product, are all faults that relate to a product and not a result of machine / operational malfunction. They are defined as products which do not fulfil the needs and expectations of the customer. As an example, the maximum current consumption of a car IC is measured by a digital multi meter. If the maximum current exceeds a level specified by the customer, it is defined as a product fault. Other examples are the brightness intensity of a pointer, or the force required to place a pointer on the motor shaft. Faults in inspection equipment on the other hand relate to faults which result in an abnormal behaviour of the inspection equipment or its capability to perform a normal operation. These occur for example when one of the sensors is faulty or when one of the actuators does not work. Configuration faults are those which are a result of either inadequate limit specification or system misconfiguration causing a part to fail despite being manufactured to the customer's requirements. Those failures very often result in line downtime, and output loss. The result for those failures is an ambiguous variation in the process. Table VI presents the percentage of correctly classified faults in each of these three categories, where a performance comparison of the proposed hybrid model with other commonly used FDI methods has been shown to assess its effectiveness. It is however important to stress the difficulty of drawing a direct comparison between the different fault detection methods. Very often the level of work involved in modifying or adding rules to rule based methods can improve the classification results. The same is true for template based methods where results can vary depending on how good the templates are, and how much expert knowledge is involved in creating the templates. Therefore it is worth looking at these comparisons from the perspective of how much human effort-time is needed to achieve different levels of recognition rate. Figure 9 shows that by using basic fault detection methods, like, for example, rule-based, better recognition results can only be achieved in the initial phase. With the complexity of the underlying process growing, consideration should be given to using methods capable of learning, to automatically identify faults from examples.

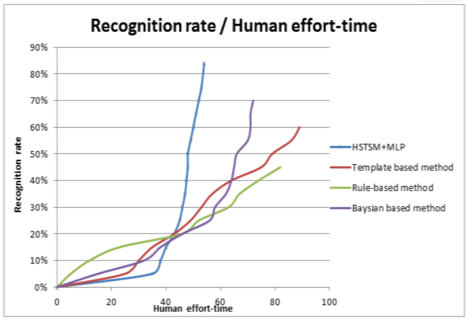

Fig. 9. Recognition rate for different fault detection methods.

The proposed approach although initially requiring more human effort (for algorithm specification and features extraction) yields better classification results later. It can be concluded that to maximise the benefits of using a fault detection system it is important to assess the complexity of the process being modelled, to determine an appropriate method that matches the complexity of the chosen problem.

To further assess the system, there was a need to measure how the fault recognition rate changed as a function of the number of training samples. To achieve this, the number of training examples presented to the model was varied and the performance on the validation set checked, see Figure 10.

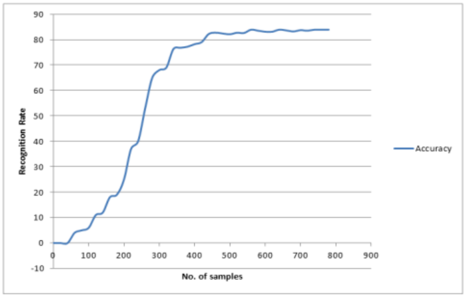

Fig. 10. Recognition rate change with change in number of training samples.

Figure 10 shows positive correlation between the number of samples used to train the model and the recognition rate for faults in a product. The recognition rate for faults in a product class steadily increases until a certain point, after which the increase in number of samples does not improve the recognition rate and starts oscillating around 84%.

V. CONCLUSIONS AND FUTURE WORK

This paper proposed a novel soft computing generative model called HSTSM. The model integrates several machine learning techniques to address the problem of automatic FDI in computer based automated inspection systems applied in the production of ICs.

The overall results show that the proposed approach proves to be a valid alternative for other FDI systems where a need for systems capable of modeling complex manufacturing and control systems arises. A future study will concentrate on improving the recognition and input prediction rates of the approach. More studies have to be conducted on the stability of the model in respect of both missing data and data with different levels of noise. Finally further work will be done to apply the HSTSM approach to other application domains such as healthcare for the prediction and diagnosis of diseases [37], [38], real stock market to analyse trends and potential undesired market behaviour as well as a number of different physical systems to predict the behaviors of those systems [39].

References

[1] W. H. Woodall, “Controversies and Contradictions in Statistical Process Control,” Journal of Quality Technology, vol. 32, no. 4, pp. 341–350, 2000.

[2] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” arXiv preprint arXiv:1409.1556

[3] G. Hinton et al, “Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups,” IEEE Signal Processing Magazine, vol. 29, no. 6, pp. 82–97, 2012.

[4] J. Schmidhuber, “Deep learning in neural networks: An overview.” in Neural networks : the official journal of the International Neural Network Society 61 pp: 85-117, 2015.

[5] Z. Zhang and J. Zhao, “A deep belief network based fault diagnosis model for complex chemical processes,” in Computers & Chemical Engineering. vol. 107, pp. 395-407, 2017.

[6] J. Hawkins and D. George "Hierarchical Temporal Memory: Concepts, Theory, and Terminology" Accessed: Aug. 21, 2014. [Online]. Available: http://www-edlab.cs.umass.edu/cs691jj/hawkins-and-george-2006.pdf

[7] Y.A. Bolotova, A.A. Druki and V.G. Spitsyn, "'License plate recognition with hierarchical temporal memory model," Strategic Technology (IFOST), 2014 9th International Forum on, pp. 136-139.

[8] S. Dang et al, “Enabling Multi-Carrier Relay Selection by Sensing Fusion and Cascaded ANN for Intelligent Vehicular Communications,” IEEE Sensors Journal, pp. 1–1, 2020.

[9] Q. Li et al, “Dual-Hop Spatial Modulation With A Relay Transmitting Its Own Information,” IEEE Transactions on Wireless Communications, pp. 1–1, 2020.

[10] K. Z. Ghafoor et al, “Millimeter-Wave Communication for Internet of Vehicles: Status, Challenges and Perspectives,” IEEE Internet of Things Journal, pp. 1–1, 2020.

[11] B. Zhou et al, “Performance Limits of Visible Light-Based Positioning for Internet-of-Vehicles: Time-Domain Localization Cooperation Gain,” IEEE Transactions on Intelligent Transportation Systems, pp. 1–15, 2020.

[12] Q. Li et al, “Subcarrier Index Modulation for Future Wireless Networks: Principles, Applications, and Challenges,” IEEE Wireless Communications, pp. 1–8, 2020.

[13] L. Birek et al, “A novel Big Data analytics and intelligent technique to predict drivers intent,” Journal of Computers in Industry, vol. 99, pp. 226–240, 2018.

[14] A. Grzywaczewski and R. Iqbal, “Task-specific information retrieval systems for software engineers,” Journal of Computer and System Sciences, vol. 78, no. 4, pp. 1204–1218, 2012.

[15] N. M. Paz, W. Leigh and R. V. Rogers, "The development of knowledge for maintenance management using simulation," in IEEE Transactions on Systems, Man, and Cybernetics, vol. 24, no. 4, pp. 574-593, April 1994.

[16] V. Venkatasubramanian, “A review of process fault detection and diagnosis Part I: Quantitative model-based methods,” in Computers & Chemical Engineering. vol. 27. pp. 293 – 311, 2003.

[17] H. Chen and B. Jiang, "A Review of Fault Detection and Diagnosis for the Traction System in High-Speed Trains," in *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 2, pp. 450-465, Feb. 2020.

[18] N. Meskin and K. Khorasani, "Fault Detection and Isolation: Multi-Vehicle Unmanned Systems" 2011.

[19] R. Isermann, "Model base fault detection and diagnosis methods," *American Control Conference, Proceedings of the 1995*, vol. 3, pp. 1605-1609 vol.3.

[20] H. Chen, B. Jiang and N. Lu, "A Newly Robust Fault Detection and Diagnosis Method for High-Speed Trains," in *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 6, pp. 2198-2208, June 2019.

[21] M. Djemai, J.P. Barbot and O. Bethoux, "'On the problem of fault detection and residual generation," *Decision and Control, 2000. Proceedings of the 39th IEEE Conference on*, vol. 5, pp. 4335-4340 vol.5

[22] R.J. Patton and J. Chen, "'Advances in fault diagnosis using analytical redundancy," *Plant Optimisation for Profit (Integrated Operations Management and Control), IEE Colloquium on (Digest No.1993/019)*, pp. 6/1-612.

[23] K. Zhang, B. Jiang, X. Yan and Z. Mao, "Incipient Fault Detection for Traction Motors of High-Speed Railways Using an Interval Sliding Mode Observer," in *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 7, pp. 2703-2714, July 2019.

[24] D. Foxvog and M. Kurki, "Survey of real-time and on-line diagnostic expert systems," *Real Time Systems, 1991. Proceedings., Euromicro '91 Workshop on*, pp. 61-69.

[25] H.R. Karimi, M. Chadli and P. Shi, "'Fault detection, isolation, andtolerant control of vehicles using soft computing methods," *Control Theory & Applications, IET*, vol. 8, no. 9, pp. 655-657.

[26] R. Iqbal et al, “Fault Detection and Isolation in Industrial Processes Using Deep Learning Approaches,” IEEE Transactions on Industrial Informatics, vol. 15, no. 5, pp. 3077–3084, 2019.

[27] J. Rowley, "'The wisdom hierarchy: representations of the DIKW hierarchy." *J.Information Science*, vol. 33, no. 2, pp. 163-180.

[28] G.E. Hinton, "'A Practical Guide to Training Restricted Boltzmann Machines.", vol. 7700, pp. 599-619.

[29] G.E. Hinton, "'Training products of experts by minimising contrastive divergence " *Neural Comput.*, vol. 14, no. 8, Aug, pp. 1771-1800.

[30] G.E. Hinton, S. Osindero and Y.W. Teh, "A fast learning algorithm for deep belief nets " *Neural Comput.*, vol. 18, no. 7, Jul, pp. 1527-1554.

[31] G.E. Hinton, "'Reducing the Dimensionality of Data with Neural Networks " *Science*, vol. 313, no. 5786, pp. 504 -507.

[32] L. Kaufman and P.J. Rousseeuw, "'Finding Groups in Data: An Introduction to Cluster Analysis ", 2005.

[33] I. Syarif, A. Prugel-Bennett and G. Wills, 2016. SVM parameter optimization using grid search and genetic algorithm to improve classification performance. Telkomnika, 14(4), p.1502.

[34] D.M. Hawkins, "'The problem of overfitting," *J.Chem.Inf.Comput.Sci.*, vol. 44, no. 1, pp. 1-12.

[35] G.E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever and R.R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*.

[36] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929-1958.

[37] J. Navarro et al, “Fuzzy adaptive cognitive stimulation therapy generation for Alzheimer’s sufferers: Towards a pervasive dementia care monitoring platform,” Future Generation Computer Systems. Elsevier, vol. 88, pp.479-490, 2018

[38] S. Mahmud, R. Iqbal, F. Doctor, “Cloud enabled data analytics and visualization framework for health-shocks prediction”, Journal of Future Generation of Computer Systems, Elsevier, Vol 65, pp. 169–181

[39] R. Iqbal et al, “Big Data analytics and Computational Intelligence for Cyber–Physical Systems: Recent trends and state of the art applications”, Future Generation Computer Systems (2017)